Terms used **profiling optimizing loops**

Found **24,091** of **201,798**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 A self-optimizing embedded microprocessor using a loop table for low power

Frank Vahid, Ann Gordon-Ross

August 2001 **Proceedings of the 2001 international symposium on Low power electronics and design ISLPED '01**

Publisher: ACM Press

Full text available: pdf(230.48 KB) Additional Information: full citation, references, citings, index terms

Keywords: cores, embedded systems, low-power, parameterized architectures, platforms, self-optimizing architecture, system-on-a-chip, tuning

2 Profile-based optimizations: Dynamic trace selection using performance monitoring hardware sampling

Howard Chen, Wei-Chung Hsu, Jiwei Lu, Pen-Chung Yew, Dong-Yuan Chen

March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization CGO '03**

Publisher: IEEE Computer Society

Full text available: pdf(1.88 MB) Additional Information: full citation, abstract, references, citings, index terms

Optimizing programs at run-time provides opportunities to apply aggressive optimizations to programs based on information that was not available at compile time. At run time, programs can be adapted to better exploit architectural features, optimize the use of dynamic libraries, and simplify code based on run-time constants.Our profiling system provides a framework for collecting information required for performing run-time optimization. We sample the performance hardware registers available on ...

3 Software for high-performance systems: Identifying potential parallelism via loop-centric profiling

Tipp Moseley, Daniel A. Connors, Dirk Grunwald, Ramesh Peri

May 2007 **Proceedings of the 4th international conference on Computing frontiers CF '07**

Publisher: ACM Press

Full text available: pdf(278.02 KB) Additional Information: full citation, abstract, references, index terms

The transition to multithreaded, multi-core designs places a greater responsibility on programmers and software for improving performance; thread-level parallelism (TLP) will be increasingly relied upon in addition to instruction-level parallelism (ILP) and increased clock frequency. Deciding where to try to parallelize code is difficult, especially for large, complex applications or those where the original developers have moved on. Outer loops are relatively easy targets for parallelization ...

Keywords: loop profiling, parallelization

4 Extending Path Profiling across Loop Backedges and Procedure Boundaries

Sriraman Tallam, Xiangyu Zhang, Rajiv Gupta

March 2004 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization CGO '04**

Publisher: IEEE Computer Society

Full text available: pdf(416.54 KB) Additional Information: full citation, abstract, citings, index terms

Since their introduction, path profiles have been used toguide the application of aggressive code optimizations andperforming instruction scheduling. However, for optimizationand scheduling, it is often desirable to obtain frequencycounts of paths that extend across loop iterations and crossprocedure boundaries. These longer paths, referred to asinteresting paths in this paper, account for over 75% of theflow in a subset of SPEC benchmarks. Although the frequencycounts of interesting paths can b ...

Keywords: path profiles, overlapping path profiles, profileguided optimization, and instruction scheduling

5 Microprocessor architecture: Frequent loop detection using efficient non-intrusive on-chip hardware

Ann Gordon-Ross, Frank Vahid

October 2003 **Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems CASES '03**

Publisher: ACM Press

Full text available: pdf(278.07 KB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic software optimization methods are becoming increasingly popular for improving software performance and power. The first step in dynamic optimization consists of detecting frequently executed code, or "critical regions." Previous critical region detectors have been targeted to desktop processors. We introduce a critical region detector targeted to embedded processors, with the unique features of being very size and power efficient, and being completely non-intrusive to the software's exec ...

Keywords: dynamic optimization, frequent loop detection, frequent value profiling, hardware profiling, hot spot detection, on-chip profiling, runtime profiling

6 Continuous program optimization: A case study

Thomas Kistler, Michael Franz

July 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 4

Publisher: ACM Press

Full text available: pdf(877.67 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Much of the software in everyday operation is not making optimal use of the hardware on which it actually runs. Among the reasons for this discrepancy are hardware/software mismatches, modularization overheads introduced by software engineering considerations, and the inability of systems to adapt to users' behaviors.A solution to these problems is to delay code generation until load time. This is the earliest point at which a piece of software can be fine-tuned to the actual capabilities of the ...

Keywords: Dynamic code generation, continuous program optimization, dynamic reoptimization

7 Practicing JUDO: Java under dynamic optimizations

Michał Cierniak, Guei-Yuan Lueh, James M. Stichnoth

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5

Publisher: ACM Press

Full text available: pdf(190.06 KB) Additional Information: full citation, abstract, references, citings, index terms

A high-performance implementation of a Java Virtual Machine (JVM) consists of efficient implementation of Just-In-Time (JIT) compilation, exception handling, synchronization mechanism, and garbage collection (GC). These components are tightly coupled to achieve high performance. In this paper, we present some static anddynamic techniques implemented in the JIT compilation and exception handling of the Microprocessor Research Lab Virtual Machine (MRL VM), ...

8 Optimally profiling and tracing programs

Thomas Ball, James R. Larus

July 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 4

Publisher: ACM Press

Full text available: pdf(2.84 MB) Additional Information: full citation, abstract, references, citings, index terms, review

This paper describes algorithms for inserting monitoring code to profile and trace programs. These algorithms greatly reduce the cost of measuring programs with respect to the commonly used technique of placing code in each basic block. Program profiling counts the number of times each basic block in a program executes. Instruction tracing records the sequence of basic blocks traversed in a program execution. The algorithms optimize the placement of counting/tracing code with respect to the ...

Keywords: control-flow graph, instruction tracing, instrumentation, profiling

9 High-level power estimation (invited talks): Source code optimization and profiling of energy consumption in embedded systems

Tajana Šimunić, Luca Benini, Giovanni De Micheli, Mat Hans

September 2000 **Proceedings of the 13th international symposium on System synthesis ISSS '00**

Publisher: IEEE Computer Society

Full text available: pdf(81.88 KB) Additional Information: full citation, abstract, references, citings

This paper presents a source code optimization methodology and a profiling tool that have been developed to help designers in optimizing software performance and energy in embedded systems. Code optimizations are applied at three levels of abstraction: algorithmic, data and instruction-level. The profiler exploits a cycle-accurate energy consumption simulator [3] to relate the embedded system energy consumption and performance to the source code. Thus, it can be used for analysis (i.e., to find ...

10 Performance of Runtime Optimization on BLAST

Abhinav Das, Jiwei Lu, Howard Chen, Jinpyo Kim, Pen-Chung Yew, Wei-Chung Hsu, Dong-Yuan Chen

March 2005 **Proceedings of the international symposium on Code generation and optimization CGO '05**

Publisher: IEEE Computer Society

Full text available: pdf(218.95 KB) Additional Information: full citation, abstract, index terms

Optimization of a real world application BLAST is used to demonstrate the limitations of static and profile-guided optimizations and to highlight the potential of runtime optimization systems. We analyze the performance profile of this application to determine performance bottlenecks and evaluate the effect of aggressive compiler optimizations on BLAST. We find that applying common optimizations (e.g. O3) can degrade performance. Profile guided optimizations do not show much improvement across t ...

11 Profiling tools for hardware/software partitioning of embedded applications

Dinesh C. Suresh, Walid A. Najjar, Frank Vahid, Jason R. Villarreal, Greg Stitt

June 2003 **ACM SIGPLAN Notices , Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems LCTES '03**, Volume 38 Issue 7

Publisher: ACM Press

Additional Information: full citation, abstract, references, citings, index

Full text available: pdf(228.38 KB) terms

Loops constitute the most executed segments of programs and therefore are the best candidates for hardware software partitioning. We present a set of profiling tools that are specifically dedicated to loop profiling and do support combined function and loop profiling. One tool relies on an instruction set simulator and can therefore be augmented with architecture and micro-architecture features simulation while the other is based on compile-time instrumentation of gcc and therefore has very litt ...

Keywords: compiler optimization, hardware/software partitioning, loop analysis

12 1 - Special Section: Link-time compaction and optimization of ARM executables

Bjorn De Sutter, Ludo Van Put, Dominique Chanet, Bruno De Bus, Koen De Bosschere
February 2007 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 6 Issue 1
Publisher: ACM Press
Full text available: pdf(636.53 KB) Additional Information: full citation, abstract, references, index terms

The overhead in terms of code size, power consumption, and execution time caused by the use of precompiled libraries and separate compilation is often unacceptable in the embedded world, where real-time constraints, battery life-time, and production costs are of critical importance. In this paper, we present our link-time optimizer for the ARM architecture. We discuss how we can deal with the peculiarities of the ARM architecture related to its visible program counter and how the introduced over ...

Keywords: Performance, compaction, linker, optimization

13 Predicting program behavior using real or estimated profiles

David W. Wall
May 1991 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1991 conference on Programming language design and implementation PLDI '91**, Volume 26 Issue 6
Publisher: ACM Press
Full text available: pdf(977.81 KB) Additional Information: full citation, references, citings, index terms

14 Compilation and dynamic optimization: Dynamic parallelization and mapping of binary executables on hierarchical platforms

Efe Yardimci, Michael Franz
May 2006 **Proceedings of the 3rd conference on Computing frontiers CF '06**
Publisher: ACM Press
Full text available: pdf(241.09 KB) Additional Information: full citation, abstract, references, index terms

As performance improvements are being increasingly sought via coarse-grained parallelism, established expectations of continued sequential performance increases are not being met. Current trends in computing point towards platforms seeking performance improvements through various degrees of parallelism, with coarse-grained parallelism features becoming commonplace in even entry-level systems.Yet the broad variety of multiprocessor configurations that will be available that differ in the number o ...

Keywords: continuous optimization, dynamic parallelization

15 The Accuracy of Initial Prediction in Two-Phase Dynamic Binary Translators

Youfeng Wu, Mauricio Breternitz, Justin Quek, Orna Etzion, Jesse Fang
March 2004 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization CGO '04**
Publisher: IEEE Computer Society
Full text available: pdf(234.04 KB) Additional Information: full citation, abstract, citings, index terms

Dynamic binary translators use a two-phase approachto identify and optimize frequently executed codedynamically. In the first step (profiling phase), blocks ofcode are

interpreted or quickly translated to collectexecution frequency information for the blocks. In thesecond phase (optimization phase), frequently executedblocks are grouped into regions and advancedoptimizations are applied on them.This approach implicitly assumes that the initialprofile of each block is representative of the blockt ...

16 Using branch handling hardware to support profile-driven optimization

Thomas M. Conte, Burzin A. Patel, J. Stan Cox

November 1994 **Proceedings of the 27th annual international symposium on Microarchitecture MICRO 27**

Publisher: ACM Press

Full text available: pdf(954.48 KB) Additional Information: full citation, abstract, references, citings, index terms

Profile-based optimizations can be used for instruction scheduling, loop scheduling, data preloading, function in-lining, and instruction cache performance enhancement. However, these techniques have not been embraced by software vendors because programs instrumented for profiling run 2-30 times slower, an awkward compile-run-recompile sequence is required, and a test input suite must be collected and validated for each program. This paper proposes using existing bran ...

17 Toward a methodology of optimizing programs for high-performance computers

Rudolf Eigenmann

August 1993 **Proceedings of the 7th international conference on Supercomputing ICS '93**

Publisher: ACM Press

Full text available: pdf(1.20 MB) Additional Information: full citation, abstract, references, citings, index terms

This report describes experiences in porting the Perfect Benchmarks programs to the Alliant FX/8 and Cedar machines. Although there is much to learn before we can really say we know how to optimize programs successfully for parallel computers in general and for hierarchical shared-memory architectures in particular, it is hoped that this report will be helpful to those who are beginning to port and transform programs for parallel machines. Perhaps it is even an interesting reference for exp ...

18 Handling irreducible loops: optimized node splitting versus DJ-graphs

Sebastian Unger, Frank Mueller

July 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 4

Publisher: ACM Press

Full text available: pdf(386.11 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper addresses the question of how to handle irreducible regions during optimization, which has become even more relevant for contemporary processors since recent VLIW-like architectures highly rely on instruction scheduling. The contributions of this paper are twofold. First, a method of optimized node splitting to transform irreducible regions of control flow into reducible regions is formally defined and its correctness is shown. This method is superior to approaches previously publishe ...

Keywords: Code optimization, compilation, control flow graphs, instruction-level parallelism, irreducible flowgraphs, loops, node splitting, reducible flowgraphs

19 Interaction cost and shotgun profiling

Brian A. Fields, Rastislav Bodik, Mark D. Hill, Chris J. Newburn

September 2004 **ACM Transactions on Architecture and Code Optimization (TACO)**, Volume 1 Issue 3

Publisher: ACM Press

Full text available: pdf(647.17 KB) Additional Information: full citation, abstract, references, citings, index terms

We observe that the challenges software optimizers and microarchitects face every day boil down to a single problem: bottleneck analysis. A bottleneck is any event or resource that contributes to execution time, such as a critical cache miss or window stall. Tasks

such as tuning processors for energy efficiency and finding the right loads to prefetch all require measuring the performance costs of bottlenecks.In the past, simple event counts were enough to find the important bottlenecks. Today, t ...

Keywords: Performance analysis, critical path, modeling, profiling

20 A study of source-level compiler algorithms for automatic construction of pre-execution code

Dongkeun Kim, Donald Yeung

August 2004 **ACM Transactions on Computer Systems (TOCS)**, Volume 22 Issue 3

Publisher: ACM Press

Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings, index terms

Pre-execution is a promising latency tolerance technique that uses one or more helper threads running in spare hardware contexts ahead of the main computation to trigger long-latency memory operations early, hence absorbing their latency on behalf of the main computation. This article investigates several source-to-source C compilers for extracting pre-execution thread code automatically, thus relieving the programmer or hardware from this onerous task. We present an aggressive profile-driven co ...

Keywords: Data prefetching, memory-level parallelism, multithreading, pre-execution, prefetch conversion, program slicing, speculative loop parallelization

ACM International Conference Proceeding Series; Vol. 37 archive

Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization

2003, San Francisco, California *March 23 - 26, 2003*

General Chairs: Richard Johnson, Transmeta; Tom Conte, NC State University
Program Chairs: Wen-mei Hwu, University of Illinois at Urbana-Champaign

Table of Contents

Pages: 36 - 47
Full text available: Pdf(1.14 MB)
Additional Information: full citation, abstract, references, citings, index terms

Jumbo: run-time code generation for Java and its applications
Sam Kamin, Lars Clausen, Ava Jarvis
Pages: 48 - 56
Full text available: Pdf(698 KB)
Additional Information: full citation, abstract, references, citings, index terms

SESSION: **Profile-based optimizations**

Reality-based optimization
Scott McFarling
Pages: 59 - 68
Full text available: Pdf(1.09 MB)
Additional Information: full citation, abstract, references, citings, index terms

Coupling on-line and off-line profile information to improve program performance

Chandra Krintz
Pages: 69 - 78
Full text available: Pdf(1.16 MB)
Additional Information: full citation, abstract, references, citings, index terms

Dynamic trace selection using performance monitoring hardware sampling
Howard Chen, Wei-Chung Hsu, Jiwei Lu, Pen-Chung Yew, Dong-Yuan Chen
Pages: 79 - 90
Full text available: Pdf(1.88 MB)
Additional Information: full citation, abstract, references, citings, index terms

Optimal and efficient speculation-based partial redundancy elimination
Qiong Cai, Jingling Xue
Pages: 91 - 102
Full text available: Pdf(1.02 MB)
Additional Information: full citation, abstract, references, citings, index terms

SESSION: **EPIC compilation**

Optimizations to prevent cache penalties for the Intel® Itanium® 2 Processor

Jean-Francois Collard, Daniel Lavery
Pages: 105 - 114
Full text available: Pdf(1.28 MB)
Additional Information: full citation, abstract, references, index terms

Optimization for the Intel® Itanium® architecture register stack
Alex Settle, Daniel A. Connors, Gerolf Hoflehner, Dan Lavery
Pages: 115 - 124
Full text available: Pdf(907 KB)
Additional Information: full citation, abstract, references, citings, index terms

Speculative register promotion using Advanced Load Address Table (ALAT)
Jin Lin, Tong Chen, Wei-Chung Hsu, Pen-Chung Yew
Pages: 125 - 134
Full text available: Pdf(892 KB)
Additional Information: full citation, abstract, references, citings, index terms

Inlining of mathematical functions in HP-UX for Itanium® 2
James W. Thomas
Pages: 135 - 144
Full text available: Pdf(784 KB)
Additional Information: full citation, abstract, references, index terms

SESSION: **Code scheduling**

Improving quasi-dynamic schedules through region slip
Francesco Spadini, Brian Fahs, Sanjay Patel, Steven S. Lumetta
Pages: 149 - 158
Full text available: Pdf(1.02 MB)
Additional Information: full citation, abstract, references, citings, index terms

Integrated prepass scheduling for a Java Just-In-Time compiler on the IA-64 architecture

Tatsushi Inagaki, Hideaki Komatsu, Toshio Nakatani
Pages: 159 - 168
Full text available: Pdf(835 KB)
Additional Information: full citation, abstract, references, citings, index terms

Predicate-aware scheduling: a technique for reducing resource constraints
Mikhail Smelyanskiy, Scott A. Mahlke, Edward S. Davidson, Hsien-Hsin S. Lee
Pages: 169 - 178
Full text available: Pdf(1.05 MB)
Additional Information: full citation, abstract, references, index terms

Phi-Predication for light-weight if-conversion
Weihaw Chuang, Brad Calder, Jeanne Ferrante
Pages: 179 - 190
Full text available: Pdf(1.19 MB)
Additional Information: full citation, abstract, references, citings, index terms

SESSION: **Code optimization - I**

Local scheduling techniques for memory coherence in a clustered VLIW processor with a distributed data cache
Enric Gibert, Jesús Sánchez, Antonio González
Pages: 193 - 203
Full text available: Pdf(1.19 MB)
Additional Information: full citation, abstract, references, citings, index terms

Compiler optimization-space exploration
Spyridon Triantafyllis, Manish Vachharajani, Neil Vachharajani, David I. August
Pages: 204 - 215
Full text available: Pdf(1.19 MB)
Additional Information: full citation, abstract, references, citings, index terms

Optimizing memory accesses for spatial computation
Mihai Budiu, Seth C. Goldstein
Pages: 216 - 227
Full text available: Pdf(1.06 MB)
Additional Information: full citation, abstract, references, citings, index terms

Optimization opportunities created by global data reordering
Gadi Haber, Moshe Klausner, Vadim Eisenberg, Bilha Mendelson, Maxim Gurevich
Pages: 228 - 237
Full text available: Pdf(1.31 MB)
Additional Information: full citation, abstract, references, citings, index terms

SESSION: **Dynamic Adaptive compilation**

Design, implementation and evaluation of adaptive recompilation with on-stack replacement

Stephen J. Fink, Feng Qian
Pages: 241 - 252
Full text available: Pdf(1.02 MB)
Additional Information: full citation, abstract, references, citings, index terms

Adaptive online context-sensitive inlining
Kim Hazelwood, David Grove
Pages: 253 - 264
Full text available: Pdf(1.06 MB)
Additional Information: full citation, abstract, references, citings, index terms

An infrastructure for adaptive dynamic optimization
Derek Bruening, Timothy Garnett, Saman Amarasinghe
Pages: 265 - 275
Full text available: Pdf(1.16 MB)
Additional Information: full citation, abstract, references, citings, index terms

Dynamic profiling and trace cache generation
Marc Berndl, Laurie Hendren
Pages: 276 - 285
Full text available: Pdf(950 KB)
Additional Information: full citation, abstract, references, index terms

SESSION: **Performance monitoring**

METRIC: tracking down inefficiencies in the memory hierarchy via binary rewriting
Jaydeep Marathe, Frank Mueller, Tushar Mohan, Bronis R. de Supinski, Sally A. McKee, Andy Yoo

Pages: 289 - 300
Full text available: Pdf(1.77 MB)
Additional Information: full citation, abstract, references, citings, index terms

TEST: a tracer for extracting speculative threads
Michael Chen, Kunle Olukotun
Pages: 301 - 312
Full text available: Pdf(1.76 MB)
Additional Information: full citation, abstract, references, citings, index terms

SESSION: **Code optimization II**

Code optimization for code compression
Milenko Drinić, Darko Kirovski, Hoi Vo
Pages: 315 - 324
Full text available: Pdf(1.07 MB)
Additional Information: full citation, abstract, references, citings, index terms

Hiding program slices for software security
Xiangyu Zhang, Rajiv Gupta
Pages: 325 - 336
Full text available: Pdf(1.05 MB)
Additional Information: full citation, abstract, references, citings, index terms

Addressing mode selection
Erik Eckstein, Bernhard Scholz
Pages: 337 - 346
Full text available: Pdf(834 KB)
Additional Information: full citation, abstract, references, index terms

Web Results 1 - 10 of about 543 for loop peeling loop unrolling pipelining data prefetching. (0.51 second

SGI TPL (IRIX 6.5: Developer/OrOn2_PfTune - Chapter 7. Using **Loop** ...
Loop peeling is the technique of removing iterations from the beginning ... A better way to reduce **prefetch** overhead is to **unroll** the **loop** as shown in ...
techpubs.sgi.com/.../cgi-bin/getdoc.cgi?coll=0650&db=bks&fname=/SGI_Developer/OrOn2_PfTune/ch07.html - 87k - Cached - Similar pages

SGI TPL (IRIX 6.4: Developer/OrOn2_PfTune - Chapter 7. Using **Loop** ...
The LNO applies **loop peeling** and then **loop** fusion to produce code like ... A better way to reduce **prefetch** overhead is to **unroll** the **loop** as shown in ...
techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi/0640/bks/SGI_Developer/books/OrOn2_PfTune/sgi_html/ch07.html - 83k - Cached - Similar pages

Intel® - Compilers Overview
New features include support for advanced, dynamic **data** alignment strategies, including **loop peeling** to generate aligned loads and **loop unrolling** to match ...
www.hiperism.com/Intel/Compilers/Reseller_ProductPage_Compiler_Overview_005.htm - 55k - Cached - Similar pages

Intel® C++ Compiler 8.1 for QNX Neutrino* RTOS - Intel® Software ...
Data prefetching inserts **prefetch** instructions for selected **data** references ... including **loop peeling** to generate aligned loads and **loop unrolling** to match ...
www.intel.com/cd/software/products/asmo-na/eng/219761.htm - 31k - Cached - Similar pages

Intel® C++ Compiler for QNX Neutrino* RTOS - Intel® Software Network
Data Prefetching: **Data prefetching** is an effective technique to hide memory ... including **loop peeling** to generate aligned loads and **loop unrolling** to match ...
www.intel.com/cd/software/products/asmo-na/eng/compilers/219708.htm - 28k - Cached - Similar pages

Mechanism for software **pipelining loop** nests - US Patent 6820250
Efficient explicit **data prefetching** analysis and code generation in a ... Compiler for performing a **loop** fusion, dependent upon **loop peeling** and/or **loop** ...
www.patentstorm.us/patents/6820250.html - 19k - Cached - Similar pages

Compiler optimization techniques for exploiting a zero overhead ...
For instance, the above-noted **loop unrolling** is a popular technique to decrease **loop** ... **Data** flow analysis and **loop peeling** may be used to avoid redundant ...
www.patentstorm.us/patents/6367071-description.html - 56k - Cached - Similar pages
[More results from www.patentstorm.us]

[PDF] Proceedings Template - WORD
File Format: PDF/Adobe Acrobat
Block **Unroll** and Jam. **Loop** Fusion. Scalar Replacement. **Data Prefetching** ... code motion, and software **pipelining**. For example, a **loop** interchange can make ...
www.springerlink.com/index/83BGJ0283KXPQX9N.pdf - Similar pages

[PPT] www.caam.rice.edu/~timwar/MA471F03/OPTma471.ppt
File Format: Microsoft Powerpoint - View as HTML
Some compilers also provide +odataprefetch to indicate that **prefetch** instructions should ... **Loop Unrolling**. This technique reduces the effect of branches, ...
Similar pages

[PDF] Continuous trip count profiling for **loop** optimizations in two ...

File Format: PDF/Adobe Acrobat

peeling but neither software **pipelining** nor **data**. **prefetching** may be applied profitably. ...
candidates for **loop unrolling** and software **pipelining** but ...
ieeexplore.ieee.org/iel5/9098/28871/01299505.pdf?arnumber=1299505 - Similar pages

Google

loop peeling loop unrolling pipelining data pref Search Advanced Search Preferences

Web Results 11 - 20 of about 543 for loop peeling loop unrolling pipelining data prefetching. (0.15 secon

Crescent Bay Software -- OEM Compiler Optimization Technology -- Menu
Reduction **unrolling**. **Loop** splitting. **Loop** fusion (jamming). **Loop peeling**. ... Optimize separate **data** memories. Insert cache line **prefetch** operations. ...
www.crescentbaysoftware.com/tech_menu2.html - 6k - Cached - Similar pages

[PDF] Impact of Compiler-based **Data-Prefetching** Techniques on SPEC OMP ...
File Format: PDF/Adobe Acrobat
Note that the conditional statements used to control **data**. **prefetching** can be removed by **loop unrolling**, strip-mining,. and **peeling**. ...
ieeexplore.ieee.org/iel5/9722/30685/01419874.pdf?arnumber=1419874 - Similar pages

[PDF] Microsoft PowerPoint - EPIC-4Presentation.ppt
File Format: PDF/Adobe Acrobat - View as HTML
synchronous **data prefetch** in cache for next **loop** iteration. • there are some losses because of ... **Loop**: **unrolling**, **peeling**, fusion, nesting, etc. ...
rogue.colorado.edu/EPIC4/presentations/k2_vol_6pclr.pdf - Similar pages

Patents in Class 717/160
65, US6148439, Nested **loop data prefetching** using inner **loop** splitting and ... and system for generating compact code for the **loop unrolling** transformation ...
www.freepatentsonline.com/CCL-717-160-p2.html - 42k - Cached - Similar pages

Aggregate bandwidth through store miss management for cache-to ...
[0005] **Prefetching** by loading the next cache line in sequence can be implemented in hardware, ... [0104] **Loop** Versioning and **Loop Peeling** for **Data** Alignment ...
www.freepatentsonline.com/20050268039.html - 69k - Cached - Similar pages

[PDF] Optimizing Applications with the Intel C++ and Fortran Compilers
File Format: PDF/Adobe Acrobat - View as HTML
include **loop peeling** and **loop unrolling**. **Loop** ... Place before a **loop** to control **data prefetching**. This is supported. when -O3 is on. ...
ftp://download.intel.com/software/products/compilers/techtopics/Compiler_Optimization_7_02.pdf - Similar pages

[PDF] Microsoft PowerPoint - Pro64-Intro-1015
File Format: PDF/Adobe Acrobat - View as HTML
Heuristics integrated with software **pipelining**. • **Loop** vector dependency info passed to CG. – **Loop Peeling**. – **Loop** Tiling. – Vector **Data Prefetching** ...
www.cs.ualberta.ca/~amaral/Pro64/Pro64-Intro-1015.pdf - Similar pages

[PDF] Overview of the Open64 Compiler Infrastructure
File Format: PDF/Adobe Acrobat - View as HTML
Transformations implemented include: **loop** fission, **loop** fusion, **unroll**. and jam, **loop** interchange, **loop peeling**, **loop** tiling and vector **data prefetching**. ...
www2.cs.uh.edu/~dragon/Documents/open64-doc.pdf - Similar pages

[PDF] Hot Chips IA64 Tutorial, part 2
File Format: PDF/Adobe Acrobat - View as HTML
software **pipelining**. • **Loop unrolling** for memory aligned paired loads. **Data** alignment. and **loop** remainder issues. • Result: high bandwidth and reduced ...
www.hotchips.org/archives/hc11/1_Sun/hc99.t2.s2.IA64tut.pdf - Similar pages

[PS] Simple and Effective Array **Prefetching** in Java

File Format: Adobe PostScript - View as Text
loop unrolling and **loop peeling** to **prefetch** the references causing. cache misses. ... **pipelining** on the innermost **loop** to begin **prefetching** the array **data** ...

Web Results 21 - 30 of about 543 for loop peeling loop unrolling pipelining data prefetching. (0.21 secon

Compiler optimization - Wikipedia, the free encyclopedia
Compilers can schedule, or reorder, instructions so that **pipeline** stalls occur ... A useful special case is **loop peeling**, which can simplify a **loop** with a ...
en.wikipedia.org/wiki/Compiler_optimization - 63k - Cached - Similar pages

[PDF] Optimizing Applications with the Intel C++ and Fortran Compilers ...
File Format: PDF/Adobe Acrobat - View as HTML
Same as /O2, plus **loop** transformations and **data prefetching**. ... include **loop peeling** and **loop unrolling**. **Loop peeling** can generate aligned loads, ...
shop.rexsoft.co.kr/prod/download.asp?path_dir=down&prod_seq=000034&filename=Compiler_Optimization... - Similar pages

Optimizing Compilers for Modern Architectures - Elsevier
The basis for all the methods presented in this book is **data** dependence, ... 13.4.2 Outer **Loop Prefetching** 13.4.3 **Loop** Interchange for Scalarization 13.4.4 ...
www.elsevier.com/wps/product/cws_home/677951 - 62k - Cached - Similar pages

[PDF] Comparing and Combining Read Miss Clustering and Software **Prefetching**
File Format: PDF/Adobe Acrobat - View as HTML
prefetching algorithms apply software **pipelining** to the in-. nermost **loop** for a given miss ... Inner-**loop unrolling** also cannot help, since increases in ...
research.ac.upc.edu/pact01/papers/s9p4.pdf - Similar pages

[PPT] research.ac.upc.edu/HPCseminar/SEM9900/dehnert.ppt
File Format: Microsoft Powerpoint - View as HTML
Heuristics integrated with software **pipelining**. Fission; Fusion; **Unroll** and jam; **Loop** interchange; **Peeling**; Tiling; Vector **data prefetching**. Parallelization ...
Similar pages

[PDF] Design and Evaluation of a Compiler Algorithm for **Prefetching**
File Format: PDF/Adobe Acrobat - View as HTML
Software-controlled **data prefetching** is a promising technique for ... suppress **peeling** or **unrolling** when the **loop** becomes too large. ...
suif.stanford.edu/papers/mowry92.pdf - Similar pages

[PDF] Recurrence analysis for effective array **prefetching** in Java
File Format: PDF/Adobe Acrobat
Then, the compiler performs **loop unrolling** and **loop peeling** ... **pipelining** on the innermost **loop** to begin **prefetching** the array **data** prior to the **loop**. ...
dx.doi.org/10.1002/cpe.851 - Similar pages

An Advanced Optimizer for the IA-64 Architecture
The conditional statements that control the **data-prefetching** policy can be removed by **loop unrolling**, strip-mining, and **peeling**. However, this may result in ...
doi.ieeecomputersociety.org/10.1109/40.888704 - Similar pages

Updated for Version 6.0 Compilers
Same as /O2, plus **loop** transformations and **data prefetching**. ... Alignment strategies include **loop peeling** and **loop unrolling**. **Loop peeling** can generate ...
cluster.krasn.ru/intel/Compiler_Optimization_6.htm - 54k - Cached - Similar pages

[PDF] The superblock: An effective technique for VLIW and superscalar ...
File Format: PDF/Adobe Acrobat
expansion, **loop unrolling**, **loop peeling**, and dependence-removing optimizations ... optimizations illustrates the need to include **data prefetching** and other ...
www.springerlink.com/index/J5N10281026R6742.pdf - Similar pages

Web Results 31 - 40 of about 543 for loop peeling loop unrolling pipelining data prefetching. (0.23 secon

Enhancing Instruction Level Parallelism Through Compiler ...
loop distribution, **loop** fusion, **loop** coalescing, **loop peeling**, and **loop unrolling** [1] software **pipelining** [3] speculative execution [4], code motion, ...
citeseer.ist.psu.edu/bringmann95enhancing.html - 34k - Cached - Similar pages

[PDF] Compiler Optimization Technique for **Data** Cache **Prefetching** Using a ...
File Format: PDF/Adobe Acrobat
for large cache line size, **peeling** and **unrolling** multiple levels ... **loop** is really much shorter than the latency to **prefetch data** ...
doi.ieeecomputersociety.org/10.1109/ICPP.1994.39 - Similar pages

[PDF] **Data Prefetching**: A Cost/Performance Analysis
File Format: PDF/Adobe Acrobat - View as HTML
prefetch also requires **loop unrolling** if the code prefetches more ... software **pipelining** for **loop** prefetches, issuing prefetches for **data** ...
cdmetcalf.home.comcast.net/papers/prefetch.pdf - Similar pages

[PDF] Recurrence analysis for effective array **prefetching** in Java
File Format: PDF/Adobe Acrobat - View as HTML
Selvidge presents profile-guided software **data prefetching** ... Then, the compiler performs **loop unrolling** and **loop peeling**. to **prefetch** the references ...
www.cs.utexas.edu/users/mckinley/papers/CCPE-2004.pdf - Similar pages

[PDF] Instruction Scheduling for the HP PA-8000
File Format: PDF/Adobe Acrobat
tics: (a) For a **loop** with **data prefetch** instructions gener- ... Like software **pipelining**, **loop**. **unrolling** can reduce the stalls due to instruction latency. ...
portal.acm.org/ft_gateway.cfm?id=243900&type=pdf - Similar pages

[PDF] Performance Programming
File Format: PDF/Adobe Acrobat - View as HTML
Loop unrolling and tiling: improves **pipelining** and register us-. age. **Loop** fusion and fission: complementary techniques. **Loop peeling** ...
cs.anu.edu.au/~Alistair.Rendell/sc02/module2b.pdf - Similar pages

[PDF] Microsoft PowerPoint - Roch_Compiler.ppt
File Format: PDF/Adobe Acrobat - View as HTML
More precise **loop unrolling** for **pipelining** and parallelization of reductions. Use of dcbz to optimize store streams. **Loop unrolling** for **prefetch** stream ...
www.spscicomp.org/ScicomP9/Presentations/CompilerUpdateArchambaultCINECA.pdf - Similar pages

[PDF] Microsoft PowerPoint - IA64 Extracting the Best Performance Linux.ppt
File Format: PDF/Adobe Acrobat - View as HTML
SWP: software **pipelining** on the Itanium. SWP: software **pipelining** on the Itanium ... **Prefetch**. **Prefetch**. **Loop** interchange. **Loop** interchange. **Unrolling** ...
www.vital-it.ch/doc/IA64BestPerformance.pdf - Similar pages

[PS] lec9.ps (mpage)
File Format: Adobe PostScript - View as Text
issue **prefetch** if predicates satised. **loop** transformations to avoid conditionals. {. **peel** or split if. P. is i = 0. {. strip-mine or **unroll** if ...
amber.cs.umd.edu/tseng/archive/cmsc732-f96/lec9.2p.ps - Similar pages

[PDF] Design and Evaluation of a Compiler Algorithm for **Prefetching**

File Format: PDF/Adobe Acrobat - View as HTML

suppress **peeling**. or **unrolling**. when the **loop** becomes too large. ... uses the same software **pipelining**. technique as selective **prefetching** ...
www.cs.cmu.edu/afs/cs/academic/class/15745-s06/web/handouts/mowry-asplos92.pdf - Similar pages

The performance of runtime data cache prefetching in a d optimization system

Jiwei Lu Chen, H. Rao Fu Wei-Chung Hsu Othmer, B. Pen-Chung Yew Dong-Yuai
Dept. of Comput. Sci. & Eng., Minnesota Univ., Twin Cities, MN, USA;

This paper appears in: **Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual International Symposium on**
Publication Date: 3-5 Dec. 2003
On page(s): 180- 190
ISSN:
ISBN: 0-7695-2043-X
INSPEC Accession Number: 7947571
Digital Object Identifier: 10.1109/MICRO.2003.1253194
Posted online: 2004-01-08 13:23:33.0

Abstract
Traditional software controlled data cache prefetching is often ineffective due to the lack o and miss address information. To overcome this limitation, we implement runtime data cac dynamic optimization system ADORE (ADaptive Object code Reoptimization). Its performa compared with static software prefetching on the SPEC2000 benchmark suite. Runtime ca shows better performance. On an Itanium 2 based Linux workstation, it can increase perfc than 20% over static prefetching on some benchmarks. For benchmarks that do not benef the runtime optimization system adds only 1%-2% overhead. We have also collected cach guide static data cache prefetching in the ORC compiler. With that information the compile avoid generating prefetches for loops that hit well in the data cache.

Index Terms
Inspec
Controlled Indexing
cache storage optimising compilers program control structures storage manag

Non-controlled Indexing
ADORE Itanium 2 Linux workstation ORC compiler SPEC2000 benchmark s adaptive object code reoptimization cache miss profiles dynamic optimization s address information runtime cache miss runtime cache prefetching runtime da prefetching software controlled data cache prefetching static data cache prefetc prefetching static software prefetching

Author Keywords
Not Available

References

No references available on IEEE Xplore.

Citing Documents

No citing documents available on IEEExplore.